IN THE CLAIMS

1. (currently amended) A computer-implemented method for analysis of executable program code, the executable program including segments of code that correspond to callable functions in source code from which the executable code was generated, comprising:

reading from the executable program code pairs of entry points and endpoints, each pair including an entry point and an endpoint that are associated with a callable function in the source code and corresponding to a segment of the executable program code; ~~and~~

generating analysis data for the functions identified by the pairs of entry points and end points;~~.~~

detecting execution of stub functions at runtime; and

bypassing analysis of stub functions.

2. (original) The method of claim 1, further comprising scanning the executable program code for selected characteristics using the pairs of entry points and endpoints.

3. (original) The method of claim 1, further comprising:

executing the program code;

detecting execution of the functions using the pairs of entry points and endpoints; and

recording selected execution characteristics of each executed function.

4. (original) The method of claim 1, wherein the executable program code includes one or more dynamic load modules, the method further comprising:

reading entry points of initializer and deinitializer functions from dynamic load modules;

pairing the entry points of the initializer and deinitializer functions with endpoints of the initializer and deinitializer functions; and

generating analysis data for the initializer and de-initializer functions identified by the pairs of entry points and end points of the initializer and deinitializer functions.

5. (original) The method of claim 4, wherein the executable program code includes a procedure lookup table (PLT) table associated with the one or more dynamic load modules, the method further comprising:

reading function entry points from the PLT;

pairing the entry points from the PLT with endpoints; and

generating analysis data for the PLT functions identified by the pairs of entry points and end points of the PLT functions.

6. (original) The method of claim 4, further comprising scanning the executable program code for selected characteristics using the pairs of entry points and endpoints.

7. (original) The method of claim 4, further comprising:

executing the program code;

detecting execution of the functions using the pairs of entry points and endpoints; and

recording selected execution characteristics of each executed function.

8. (original) The method of claim 4, wherein the program code includes a symbol table identifying one or more function entry points, the method further comprising:

reading entry points of functions from the symbol table;

pairing the entry points from the symbol table with endpoints; and

generating analysis data for the symbol table functions identified by the pairs of entry points and end points of the symbol table functions.

9. (original) The method of claim 1, wherein the program code includes a symbol table identifying one or more function entry points, the method further comprising:

reading entry points of functions from the symbol table;

pairing the entry points from the symbol table with endpoints; and

generating analysis data for the symbol table functions identified by the pairs of entry points and end points of the symbol table functions.

10. (original) The method of claim 1, further comprising:

detecting function calls at runtime;

finding the entry point of a runtime-detected function call;

pairing an endpoint with the entry point of a runtime-detected function call; and

generating analysis data for functions identified by pairs of entry points and end points of the runtime-detected function calls.

11. (canceled)

12. (canceled)

13. (original) The method of claim 10, wherein the executable program code includes one or more dynamic load modules, the method further comprising:

reading entry points of initializer and deinitializer functions from dynamic load modules;

pairing the entry points of the initializer and deinitializer functions with endpoints of the initializer and deinitializer functions; and

generating analysis data for the initializer and de-initializer functions identified by the pairs of entry points and end points of the initializer and deinitializer functions.

14. (original) The method of claim 13, wherein the executable program code includes a procedure lookup table (PLT) table associated with the one or more dynamic load modules, the method further comprising:

reading function entry points from the PLT;

pairing the entry points from the PLT with endpoints; and

generating analysis data for the PLT functions identified by the pairs of entry points and end points of the PLT functions.

15. (currently amended) An apparatus for analysis of executable program code, the executable program including segments of code that correspond to callable functions in source code from which the executable code was generated, comprising:

means for reading from the executable program code pairs of entry points and endpoints, each pair including an entry point and an endpoint that are associated with a callable function in the source code and corresponding to a segment of the executable program code; ~~and~~

means for generating analysis data for the functions identified by the pairs of entry points and end points~~.~~;

means for detecting execution of stub functions at runtime; and

means for bypassing analysis of stub functions.

16. (new) A method for determining pairs of function entry points and corresponding function endpoints in executable program code for analysis of the program code, comprising:

searching in the executable program code for checkpoint description data, the checkpoint description data including associated pairs of entry points and endpoints, and storing the associated pairs of entry points and endpoints in a working set of entry points and endpoints;

searching for function entry points in a symbol table associated with the executable program code and storing the function entry points in the working set;

for each stored unpaired function entry point not having a stored associated endpoint in the working set, determining an associated endpoint based on a function entry point that follows the unpaired function entry point in the executable code, and storing in the working set each determined endpoint in association with the unpaired entry point; and

generating analysis data for the functions identified by the working set of entry points and end points.

17. (new) The method of claim 16, wherein the executable program code includes one or more dynamic load modules, the method further comprising:

reading entry points of initializer and deinitializer functions from the dynamic load modules;

determining from the dynamic load modules endpoints of the initializer and deinitializer functions associated with the entry points of the initializer and deinitializer functions; and

storing in the working set the entry points and associated endpoints of the initializer and deinitializer functions.

18. (new) The method of claim 17, wherein the executable program code has an associated procedure lookup table (PLT) table that is associated with the one or more dynamic load modules, the method further comprising reading function entry points from the PLT and storing the function entry points in the working set.

19. (new) The method of claim 16, further comprising:

executing the program, and during execution performing the steps including,

detecting execution of a branch instruction;

identifying as a potential function entry point a target address of the branch instruction;

determining whether executable code that follows the target address returns control to an instruction that immediately follows the branch instruction;

storing the potential function entry point in the working set in response to determining that the executable code that follows the target address returns control to an instruction that immediately follows the branch instruction and the function entry point not being present in the working set; and

determining an associated endpoint of the potential function entry point based on a function entry point that follows the potential function entry point in the executable code, and storing in the working the endpoint in association with the potential function entry point.

20. (new) A program storage medium, comprising:

at least one processor-readable program storage device configured with instructions executable by one or more processors for determining pairs of function entry points and corresponding function endpoints in executable program code for analysis of the program code by performing the steps including,

searching in the executable program code for checkpoint description data, the checkpoint description data including associated pairs of entry points and endpoints, and storing the associated pairs of entry points and endpoints in a working set of entry points and endpoints;

searching for function entry points in a symbol table associated with the executable program code and storing the function entry points in the working set;

for each stored unpaired function entry point not having a stored associated endpoint in the working set, determining an associated endpoint based on a function entry point that follows the unpaired function entry point in the executable code, and storing in the working set each determined endpoint in association with the unpaired entry point; and

generating analysis data for the functions identified by the working set of entry points and end points.

21. (new) The program storage medium of claim 20, wherein the executable program code includes one or more dynamic load modules, and the processor-readable program storage device is further configured with instructions executable by the processor for performing the steps including,

reading entry points of initializer and deinitializer functions from the dynamic load modules;

determining from the dynamic load modules endpoints of the initializer and deinitializer functions associated with the entry points of initializer and deinitializer functions; and

storing in the working set the entry points and associated endpoints of the initializer and deinitializer functions.

22. (new) The program storage medium of claim 21, wherein the executable program code has an associated procedure lookup table (PLT) table that is associated with the one or more dynamic load modules, and the processor-readable program storage device is further configured with instructions executable by the processor for performing the steps including reading function entry points from the PLT and storing the function entry points in the working set.

23. (new) The program storage medium of claim 20, and the processor-readable program storage device is further configured with instructions executable by the processor for performing the steps including,

executing the program, and during execution performing the steps including,

detecting execution of a branch instruction;

identifying as a potential function entry point a target address of the branch instruction;

determining whether executable code that follows the target address returns control to an instruction that immediately follows the branch instruction;

storing the potential function entry point in the working set in response to determining that the executable code that follows the target address returns control to an instruction that immediately follows the branch instruction and the function entry point not being present in the working set; and

determining an associated endpoint of the potential function entry point based on a function entry point that follows the potential function entry point in the executable code, and storing in the working the endpoint in association with the potential function entry point

24. (new) An apparatus for determining pairs of function entry points and corresponding function endpoints in executable program code for analysis of the program code, comprising:

means for searching in the executable program code for checkpoint description data, the checkpoint description data including associated pairs of entry points and endpoints, and storing the associated pairs of entry points and endpoints in a working set of entry points and endpoints;

means for searching for function entry points in a symbol table associated with the executable program code and storing the function entry points in the working set;

means, responsive to each stored unpaired function entry point not having a stored associated endpoint in the working set, for determining an associated endpoint based on a function entry point that follows the unpaired function entry point in the executable code, and storing in the working set each determined endpoint in association with the unpaired entry point; and

means for generating analysis data for the functions identified by the working set of entry points and end points.

25. (new) The apparatus of claim 24, wherein the executable program code includes one or more dynamic load modules, further comprising:

means for reading entry points of initializer and deinitializer functions from the dynamic load modules;

means for determining from the dynamic load modules endpoints of the initializer and deinitializer functions associated with the entry points of initializer and deinitializer functions; and

means for storing in the working set the entry points and associated endpoints of the initializer and deinitializer functions.

26. (new) The apparatus of claim 25, wherein the executable program code has an associated procedure lookup table (PLT) table that is associated with the one or more

dynamic load modules, the apparatus further comprising means for reading function entry points from the PLT and storing the function entry points in the working set.

27. (new) The apparatus of claim 24, further comprising:

means for executing the program;

means for detecting execution of a branch instruction during execution of the program;

means for identifying, during execution of the program, as a potential function entry point a target address of the branch instruction;

means for determining, during execution of the program, whether executable code that follows the target address returns control to an instruction that immediately follows the branch instruction;

means for storing, during execution of the program, the potential function entry point in the working set, responsive to determining that the executable code that follows the target address returns control to an instruction that immediately follows the branch instruction and the function entry point not being present in the working set; and

means for determining, during execution of the program, an associated endpoint of the potential function entry point based on a function entry point that follows the potential function entry point in the executable code, and storing in the working the endpoint in association with the potential function entry point.